6. Funktionen, Parameterübergabe

Themen dieses Kapitels:

- Begriffe zu Funktionen und Aufrufen
- Parameterübergabearten
  call-by-value, call-by-reference, call-by-value-and-result
  in verschiedenen Sprachen

Vorlesung Grundlagen der Programmiersprachen SS 2010 / Folie 601

Ziele:
Übersicht zu diesem Kapitel

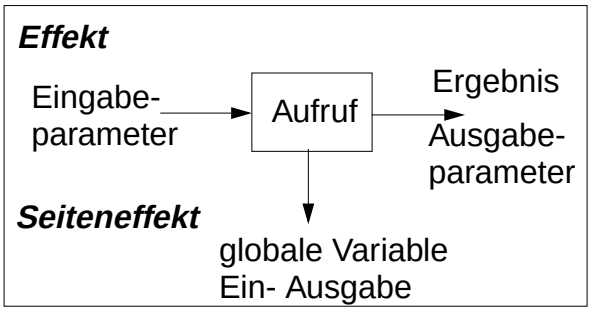
in der Vorlesung:
Erläuterungen dazu

Begriffe zu Funktionen und Aufrufen

Funktionen sind Abstraktionen von Rechenvorschriften.
Funktionen, die kein Ergebnis liefern, nennt man auch **Prozeduren**.
In objektorientierten Sprachen nennt man Funktionen auch **Methoden**.

Effekte eines Funktionsaufrufes:
Berechnung des **Funktionsergebnis** und ggf. der **Ausgabeparameter** aus den **Eingabeparametern**.

Seiteneffekte:
globale Variable schreiben,
Ein- und Ausgabe



Formale Parameter (FP): Namen für Parameter in der Funktionsdefinition.
Aktuelle Parameter (AP): Ausdrücke im Aufruf, deren Werte oder Stellen übergeben werden.

```
int Sqr (int i) { return i*i; }     Sqr (x+y)
```

Verschiedene Arten der Parameterübergabe:

call-by-value, call-by-reference, call-by-result, call-by-value-and-result, (call-by-name)

Vorlesung Grundlagen der Programmiersprachen SS 2010 / Folie 602

Ziele:

Grundbegriffe der Funktionsaufrufe

in der Vorlesung:

- Erläuterung der Begriffe

nachlesen:

..., Abschnitt 5, 5.1, 5.2

Verständnisfragen:

- Geben Sie Beispiele für Klassenmethoden in Java mit und ohne Seiteneffekt.
- Warum haben Objektmethoden i.a. Seiteneffekte?

Ausführung eines Funktionsaufrufes

Das Prinzip der Funktionsaufrufe ist in fast allen Sprachen gleich:

Ein Aufruf der Form `Funktionsausdruck (aktuelle Parmeter)`

wird in **3 Schritten** ausgeführt

1. **Funktionsausdruck auswerten**, liefert eine Funktion
2. **Aktuelle Parameter** auswerten und **an formale Parameter der Funktion binden** nach den speziellen Regeln der Parameterübergabe; Schachtel auf dem Laufzeitkeller bilden.
3. Mit diesen Bindungen den **Rumpf der Funktion ausführen** und ggf. das Ergebnis des Aufrufes berechnen; Schachtel vom Laufzeitkeller entfernen.

Beispiel:

```
z =  a[i].next.m   (x*y, b[j])
```

1. liefert Funktion
2. liefert zwei AP-Werte, werden an FP gebunden
3. Ausführung des Funktionsrumpfes liefert ein Ergebnis

Vorlesung Grundlagen der Programmiersprachen SS 2010 / Folie 602a

Ziele:

Funktionsaufrufe verstehen

in der Vorlesung:

- Erläuterung der Schritte am Beispiel

nachlesen:

..., Abschnitt 5, 5.1, 5.2

Beispiel zur Parameterübergabe

```
program
   i: integer;
   a: array [1..6] of integer;

   procedure p (x: integer, y: integer)
      t: integer;
      begin
         output x, y;          /* 2 formale Param. wie übergeben */
         t := x; x := y; y := t;
         output x, y;          /* 3 formale Param. nach Zuweisungen */
         output i, a[i];       /* 4 globale Variable der akt. Param.*/
      end;

begin
   i:= 3; a[3] := 6; a[6] := 10;
   output i, a[3];           /* 1 aktuelle Param. vor Aufruf */
   p (i, a[i]);
   output i, a[3];           /* 5 aktuelle Param. nach Aufruf */
end
```


Vorlesung Grundlagen der Programmiersprachen SS 2010 / Folie 603

Ziele:

Parameterübergabe variieren, Wirkung unterscheiden

in der Vorlesung:

- Beispiel für folgende Folien
- Programmpositionen in Bezug auf die Parameter erläutern

Call-by-value

Der **formale Parameter ist eine lokale Variable**, die **mit dem Wert des aktuellen Parameters initialisiert** wird.

Zuweisungen im Funktionsrumpf haben keine Wirkung auf die aktuellen Parameter eines Aufrufes.

Die **Werte der aktuellen Parameter** werden in die Parametervariablen **kopiert**.

Sprachen: fast alle Sprachen, z. B. Java, C, C++, Pascal, Modula-2, Ada, FORTRAN

Variante **call-by-strict-value:**
Der formale Parameter ist ein Name für den Wert des aktuellen Parameters.

Zuweisungen im Funktionsrumpf an formale Parameter sind nicht möglich.

Implementierung:

a. wie call-by-value und Zuweisungen durch Übersetzer verbieten

b. wie call-by-reference und Zuweisungen durch Übersetzer verbieten; erspart Kopieren

Sprachen: Algol-68, funktionale Sprachen

Vorlesung Grundlagen der Programmiersprachen SS 2010 / Folie 604

Ziele:

Übergabeart präzise erklären können

in der Vorlesung:

- Allgemeinste, gebräuchlichste Übergabeart
- Ausgabe des Beispiels zeigen
- call-by-strict-value verdeutlichen

nachlesen:

..., Abschnitt 5.2.1, 5.2.2

Call-by-reference

Der **formale Parameter** ist ein **Name für die Stelle des aktuellen Parameters**. Sie wird zum Zeitpunkt des Aufrufs bestimmt.

geeignet für Eingabe- und Ausgabeparameter (**transient**)
Der **aktuelle Parameter muss eine Stelle haben**: unzulässig: `h (5)` oder `h (i+1)`
Stelle des Elementes `a[i]` wird bei Beginn des Aufrufes bestimmt: `h (a[i])`
Jede **Operation mit dem formalen Parameter wirkt sofort auf den aktuellen Parameter**.

Aliasing: Mehrere Namen für dieselbe Variable (aktueller und formaler Parameter)

Vorsicht bei mehreren gleichen aktuellen Parametern! `g (x, x)`

Implementierung:
Der formale Parameter wird eine Referenzvariable. Sie wird bei einem Aufruf initialisiert mit der Stelle des aktuellen Parameters. Bei jedem Zugriff wird einmal zusätzlich dereferenziert.

Sprachen: Pascal, Modula-2, FORTRAN, C++

Vorlesung Grundlagen der Programmiersprachen SS 2010 / Folie 605

Ziele:

Zusammenhang zu Variablen und Stellen verstehen

in der Vorlesung:

- Ausgabe des Beispiels zeigen

nachlesen:

..., Abschnitt 5.2.1, 5.2.2

Verständnisfragen:

- Wie simulieren Sie call-by-reference in C?

Call-by-result

Der formale Parameter ist eine **lokale, nicht initialisierte Variable**. Ihr Wert wird **nach erfolgreichem Abarbeiten des Aufrufes an die Stelle des aktuellen Parameters zugewiesen**. Die Stelle des aktuellen Parameters wird beim Aufruf bestimmt.

Geeignet als **Ausgabeparameter**.
Die Wirkung auf den aktuellen Parameter tritt erst beim Abschluss des Aufrufs ein.
Aktueller Parameter muss eine Stelle haben.
Kopieren erforderlich.
Sprachen: Ada (`out`-Parameter)

Call-by-value-and-result

Der formale Parameter ist eine **lokale Variable, die mit dem Wert des aktuellen Parameters initialisiert wird**. Ihr Wert wird nach erfolgreichem Abarbeiten des Aufrufes an die Stelle des aktuellen Parameters zugewiesen. Die Stelle des aktuellen Parameters wird beim Aufruf bestimmt.

Geeignet als Ein- und Ausgabeparameter (**transient**);
Die Wirkung auf den aktuellen Parameter tritt erst beim Abschluss des Aufrufs ein.
Aktueller Parameter muss eine Stelle haben.
Zweimal Kopieren erforderlich.

Sprachen: Ada (`in out`-Parameter)

Vorlesung Grundlagen der Programmiersprachen SS 2010 / Folie 606

Ziele:

Unterschied zu call-by-reference verstehen

in der Vorlesung:

- Ausgabe des Beispiels zeigen

nachlesen:

..., Abschnitt 5.2.1, 5.2.2

Verständnisfragen:

- Skizzieren Sie ein möglichst kurzes Programm, das mit call-by-value-and-result oder call-by-reference unterschiedliche Ausgabe erzeugt.

Parameterübergabe in verschiedenen Sprachen

Java: nur call-by-value (auch Objektreferenzen werden call-by-value übergeben)

Pascal, Modula-2, C++ wahlweise call-by-value, call-by-reference

C#: wahlweise call-by-value, call-by-reference, call-by-result

C: nur call-by-value;
call-by-reference kann simuliert werden durch die Übergabe von Stellen:

```
void p (int i, int *a) { ... *a = 42; ... }  int x;   p (5, &x);
```

Ada: wahlweise call-by-value (**`in`**), call-by-result (**`out`**), call-by-value-and-result (**`in out`**).
Bei zusammengesetzten Objekten ist für **`in out`** auch call-by-reference möglich.
Aktuelle Parameter können auch mit den Namen der formalen benannt und dann in beliebiger Reihenfolge angegeben werden: `p (a => y[k], i => 5)`.
Für formale Parameter können default-Werte angegeben werden; dann kann der aktuelle Parameter weggelassen werden.

FORTRAN:
call-by-value, falls an den formalen Parameter nicht zugewiesen wird,
sonst call-by-reference oder call-by-value-and-result (je nach Übersetzer)

Algol-60: call-by-value, call-by-name (ist default!)
Algol-68: call-by-strict-value

funktionale Sprachen: call-by-strict-value oder lazy-evaluation (entspricht call-by-name)

Vorlesung Grundlagen der Programmiersprachen SS 2010 / Folie 607

Ziele:

Parameterübergabe wichtiger Sprachen kennen

in der Vorlesung:

- Erläuterungen dazu
- Beispiele zu Besonderheiten in Ada

Verständnisfragen:

- Welche Übergabearten sind in einer Sprache sinnvoll definierbar, wenn sie keine Variablen mit Zuweisungen hat?

Zusammenfassung zum Kapitel 6

Mit den Vorlesungen und Übungen zu Kapitel 6 sollen Sie nun Folgendes können:

- Funktionen, Aufrufen und Parameterübergabe präzise mit treffenden Begriffen erklären können
- Die Arten der Parameterübergabe unterscheiden und sinnvoll anwenden können
- Die Parameterübergabe wichtiger Sprachen kennen

Ziele:

Ziele des Kapitels erkennen

in der Vorlesung:

Erläuterungen dazu